Fast event generation on GPU

Junichi Kanzaki (KEK)

GPU Computing in High Energy Physics

Sep. 10, 2014, Univ. of Pisa, Italy

Contents

- Introduction
- Computation of Cross Sections on GPU
- VEGAS and BASES / SPRING
- Integration to MG5
- Summary & Prospects

Introduction

Motivation

- The mount of LHC data is increasing.
  - $5fb^{-1}$ in 2011 -> $22fb^{-1}$ in 2012
- High statistics data
  -> Reduction of systematic errors becomes essential for precise physics measurements.
- Better understandings of backgrounds from QCD multi-jet productions
  -> Fast event generation by changing model parameters

Overview

- Since the beginning of 2008, we have been working on the development of the code to compute amplitude/cross sections of physics processes on GPU.
- Making use of high level parallelism of GPU, we intend to improve the performance of the amplitude/cross section computations.
- We converted the FORTRAN HELAS code into the CUDA code (HEGET) which can be executed on the NVIDIA's GPU.
- Basic test of results and performance of the GPU computation was done with the QED (n-photon) and QCD (n-jet) production processes at the LHC energy.

Overview (cont'd)

- GPU versions of Monte-Carlo integration and event generation packages, BASES/SPRING (VEGAS), were developed and their performances were tested by SM processes using HEGET.
- We converted the FORTRAN HELAS code into the CUDA code (HEGET) which can be executed on the NVIDIA's GPU.
- Test of fast simulation code, PGS, was done on GPU.
- Integration to MG5: HEGET -> ALOHA generated code. Preparing interface for the SM processes.

Bibliography

- QED: K. Hagiwara, J. Kanzaki, N. Okamura, D. Rainwater and T. Stelzer, Eur. Phys. J. C66 (2010) 477, e-print arXiv:0908.4403.
- QCD: K. Hagiwara, J. Kanzaki, N. Okamura, D. Rainwater and T. Stelzer, Eur. Phys. J. C70 (2010) 513, e-print arXiv:0909.5257.
- MC integration (BASES & VEGAS): J. Kanzaki, Eur. Phys. J. C71 (2011) 1559, e-print arXiv:1010.2107.
- SM: K. Hagiwara, J. Kanzaki, Q. Li, N. Okamura, T. Stelzer, Eur.Phys.J. C73 (2013) 2608 (2013), e-print arXiv:1305.0708v2.
- Event generation (SPRING): in preparation

Our Computing Environment

	C2075	GTX580	GTX285	GTX280	9800GTX
Streaming Processors	448	512	240	←	128
Global Memory	5.4GB	1.5GB	2GB	1GB	500MB
Constant Memory	64KB	64KB	64KB	←	64KB
Shared Memory/block	48KB	48KB	16KB	←	16KB
Registers/block	32768	32768	16384	←	8192
Warp Size	32	32	32	←	32
Clock Rate	1.15GHz	1.54GHz	1.30GHz	←	1.67GHz

- NVDIA GPUs + CUDA (v4 and earlier)
- C2075: 1.03 TFLops (single), 515 GFlops (double)
- All CPU programs compared with GPU run with single core.

Computation of Cross Sections on GPU

Test with QED and QCD process

- Test with simple final states:
  - n-photon production (QED)
  - n-jet production (QCD)
- Development of basic components to calculate cross sections on GPU (CUDA)
  - Amplitude calculation:
    Heget (based on HELAS in FORTRAN)
  - Phase space generation
  - Random number generation

* Simple event loop program to calculated cross sections

Test with QED and QCD process

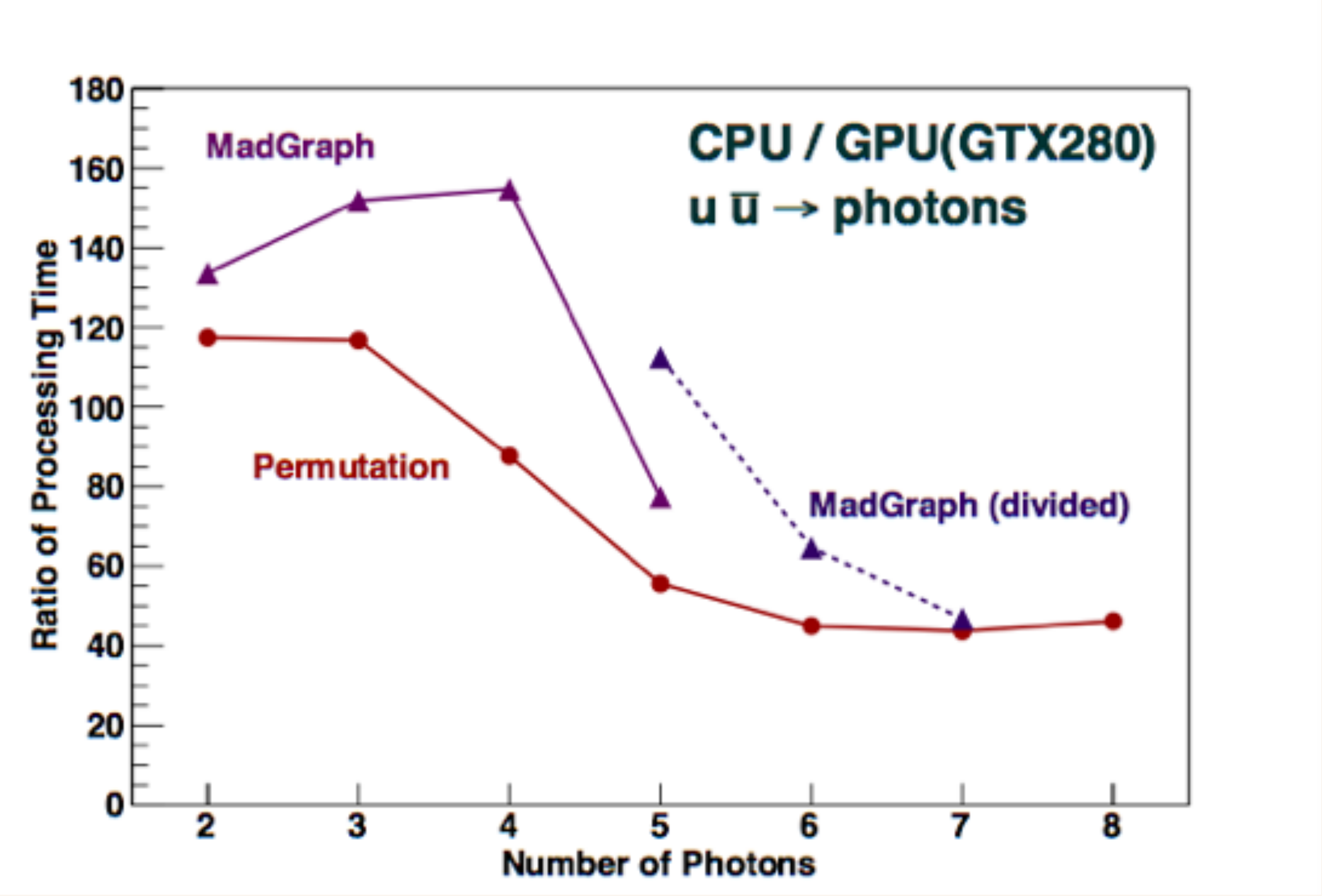
- Calculated cross sections are checked by MadGraph.
- Compare process time / loop between CPU and GPU.
- Learned and experienced "GPU computation":
  - double/single performance ratio
  - parameter dependence of performance: register allocation, no.of threads/block
  - loop unrolling

QED Processes

- uu~ -> n-photons
- Test with two kinds of amplitude:
  - MadGraph amplitude in FORTRAN -> C/CUDA
  - Amplitude written by a loop with permutations of photons (short)
- Divide a long amplitude program into smaller pieces -> successive kernel calls

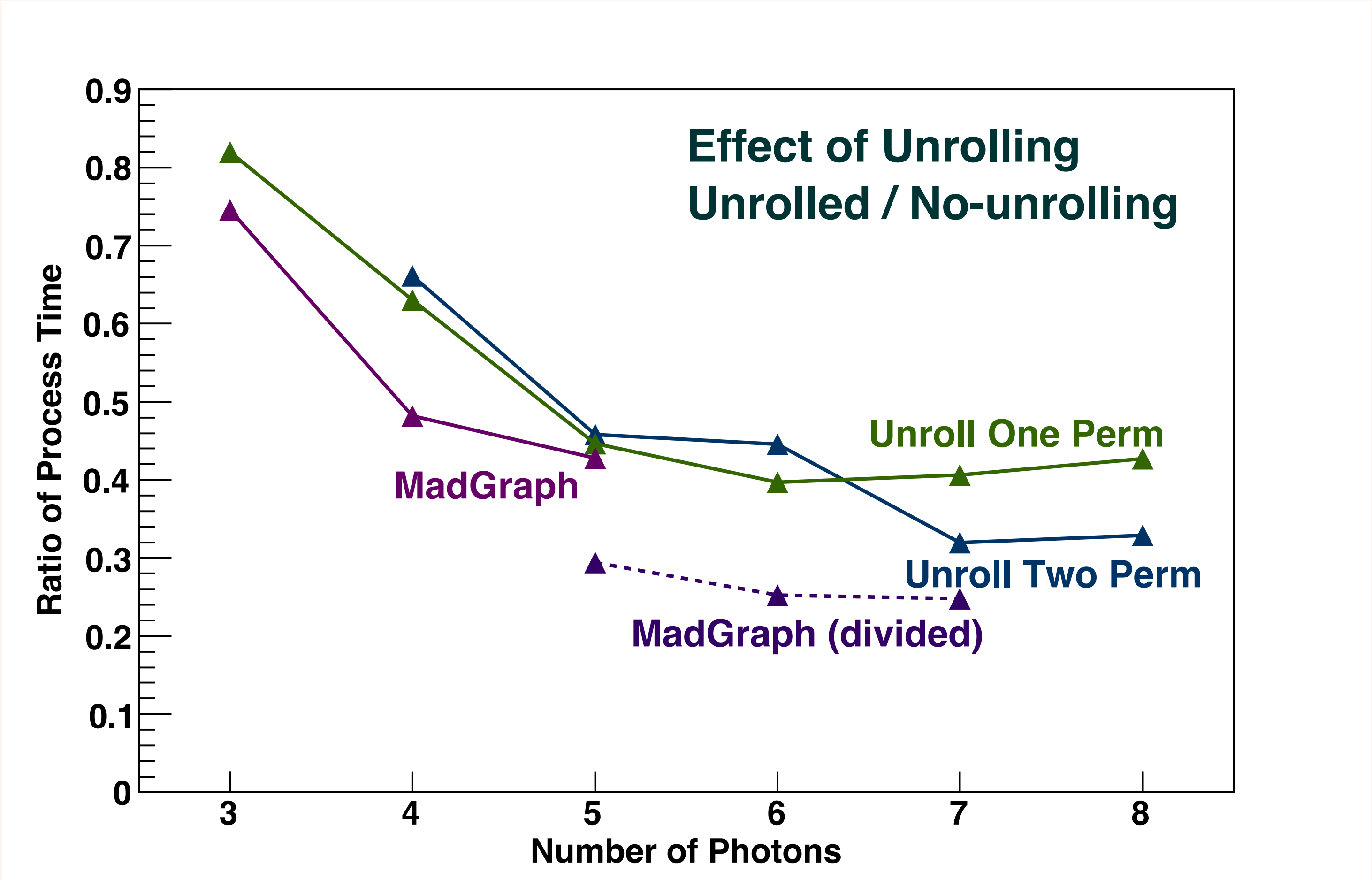
# photons	# diagrams = (# photons)!
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320

Event process time ratio (QED)



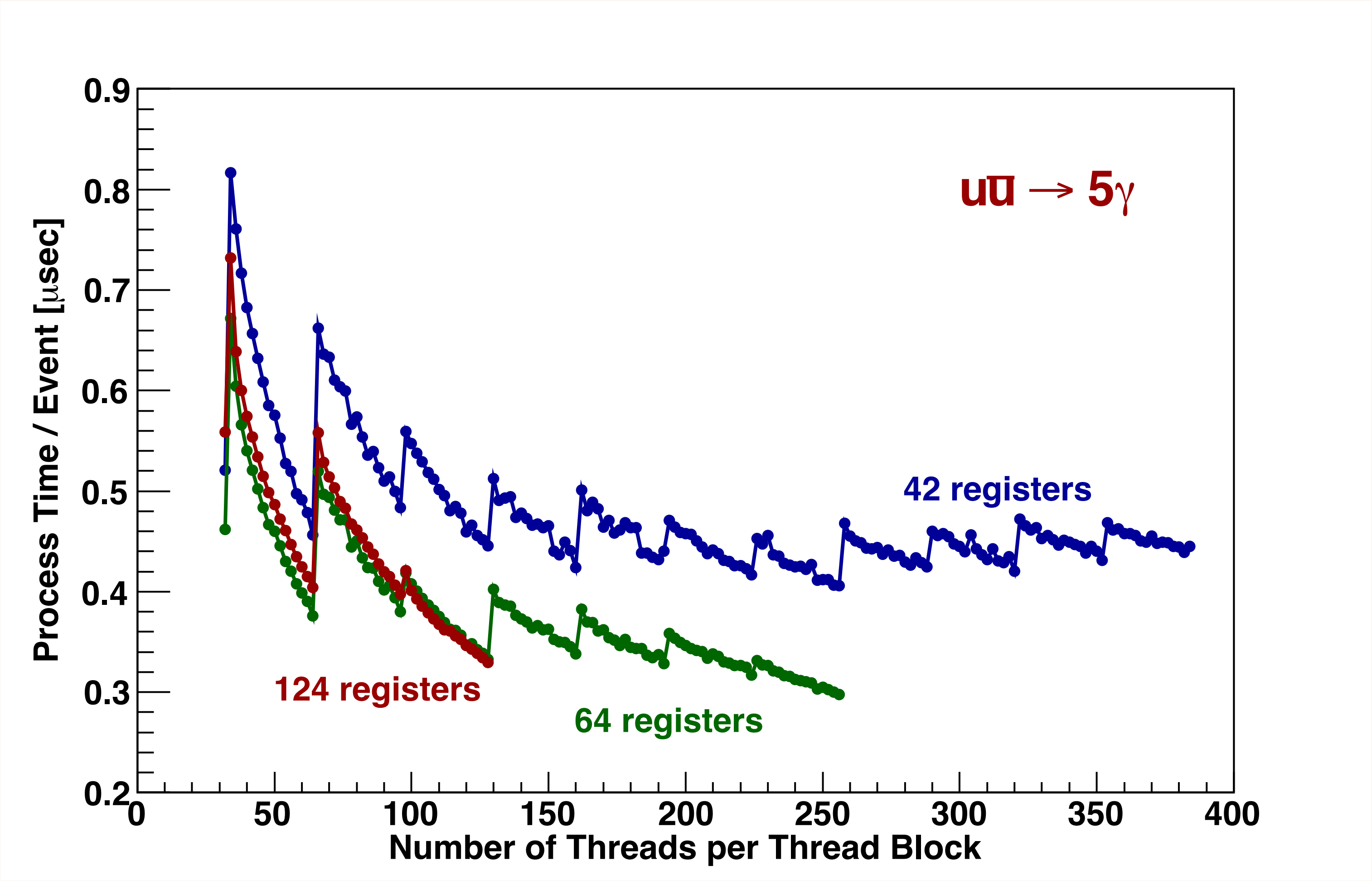
- Large reduction of process time / event loop from CPU to GPU (single precision)

Effect of Unrolling Loops



- The loop amplitude program with permutations
- Process time ratio to the no-unrolling program

Registers and thread block (GTX280)



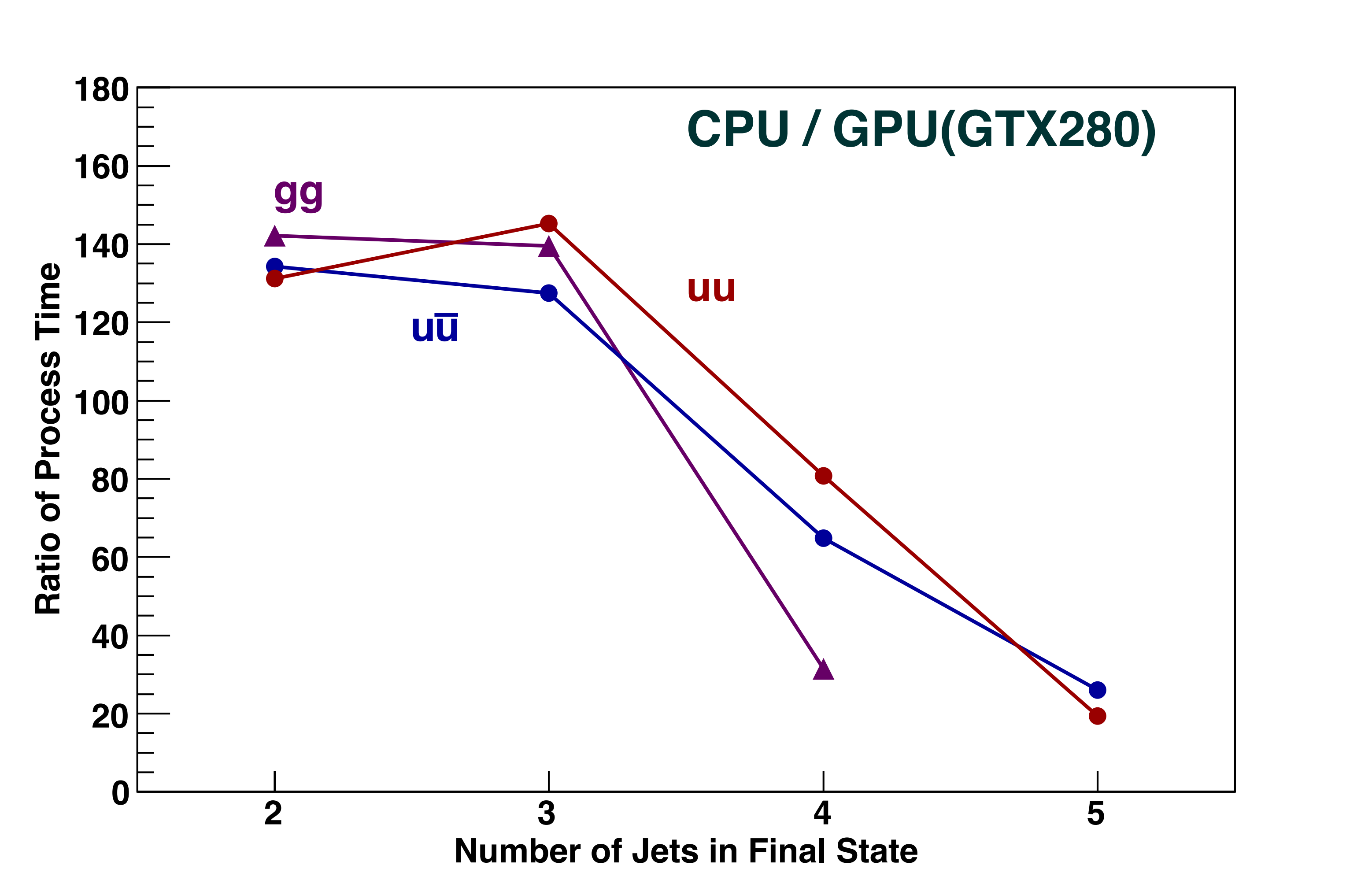
- 5-photon amplitude by MG4
- Performance depends on register allocation and thread block size (multiples of 32)

QCD Processes

# final jets	gg		uu~		uu	
	#diagram	#color	#diagram	#color	#diagram	#color
2	6	6	3	2	2	2
3	45	24	18	6	10	8
4	510	120	159	24	76	40
5	7245	720	1890	120	786	240

- uu~>n-gluons, gg>n-gluons, uu>uu+gluons
- gg>5g: the program cannot be executed on GPU.

Ratio of Process Time (QCD)



- Performance is degraded due to the size of amplitude and color factor multiplications.

BASES / SPRING
and
VEGAS

Monte Carlo integration on GPU

- For the practical event generation on GPU
  -> GPU versions of BASES/SPRING
- Application of GPU to MC integration:
  each GPU thread evaluates function value at one space point.

- Test of BASES programs using SM processes with decaying massive particles.
- Compare total process time of original FORTRAN on CPU and CUDA on GPU, and cross sections between MG5 and BASES (CPU and GPU).

Test with SM Processes

- We started more practical tests with SM processes using BASES.
- Includes decay of all massive particles:
  W>l(e, μ)ν, Z->ll (e, μ), t->W(lν)b, H->ττ
- Automatic conversion of MadGraph amplitude matrix.f -> CUDA functions (MG2CUDA)
  -> conversion program is now written in Python.
- We fixed kernel parameters:
  no. of register=64 and thread block size=256
- Double precision computations

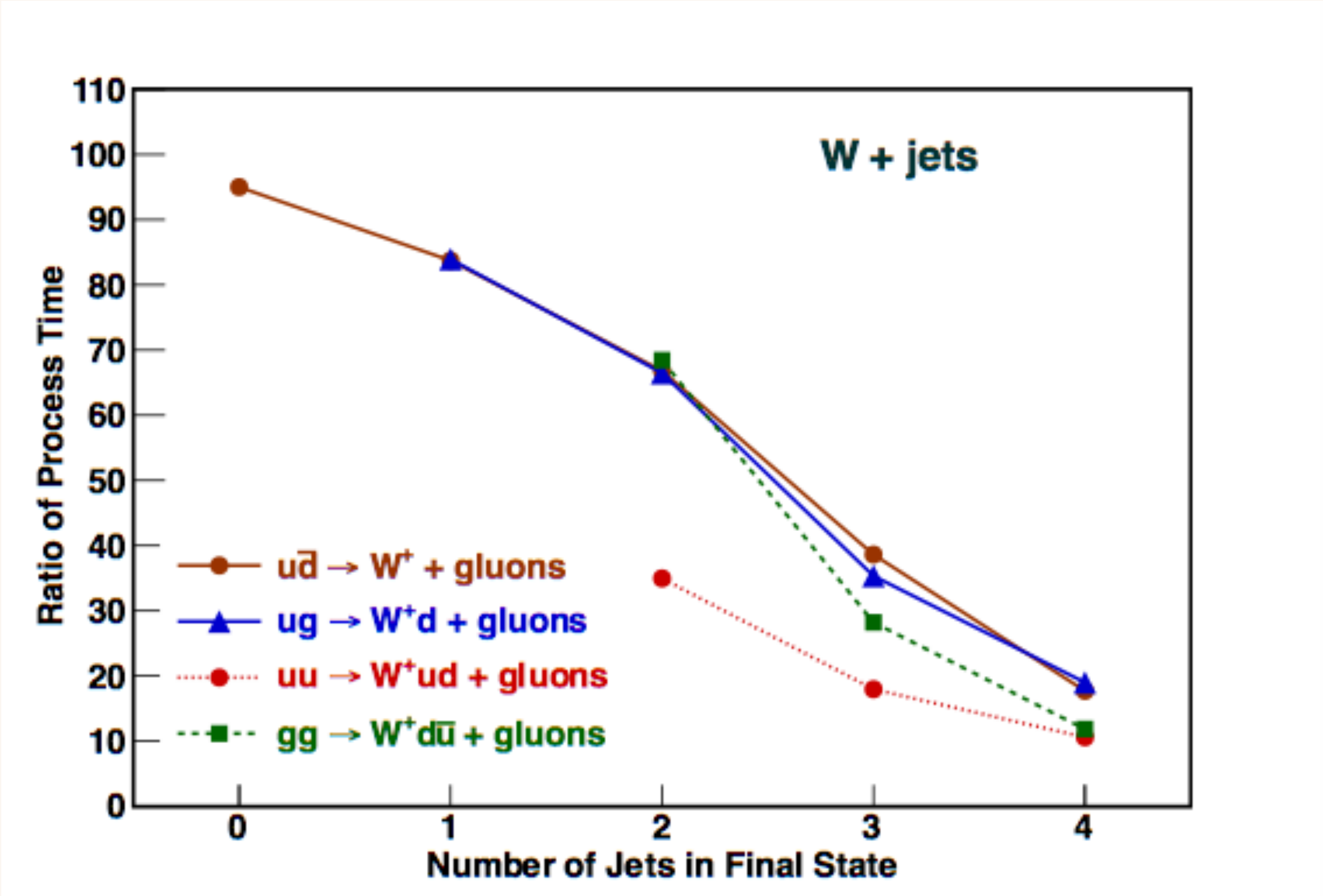
Test with SM Processes (cont'd)

- W, Z + up to 4jets:
  - ud~>W^+, ug>W^+d, uu>W^+ud, gg->W^+du~
  - uu~>Z, ug>Zu, uu>Zuu, gg>Zuu~
- WW, WZ, WW + up to 3jets:
  - uu~>W^+W^-, ug>W^+W^-u, uu>W^+W^-uu, uu>W^+W^+dd, gg->W^+W^-uu~
  - ud~>W^+Z, ug>W^+Zd, uu>W^+Zud, gg>W^+Zdu~
  - uu~>ZZ, ug>ZZd, uu->ZZuu, gg>WWuu~
- tt~+up to 3jets: uu~>tt~, ug>tt~u, uu>tt~uu, gg>tt~

Test with SM Processes (cont'd)

- HW,HZ+up to 3jets:
  - ud~>HW$^+$, ug>HW$^+$d, uu>HW$^+$ud, gg>HW$^+$du~
  - uu~>HZ, ug>HZu, uu>HZuu, gg>HZuu~
- Httx+2jets: uu~>Htt~, ug>Htt~u, uu>Htt~uu, gg>Htt~
- H(WBF)+2jets: ud>Hud, uu>Huu, ug>Hudd~, gg>Huu~dd~
- HH+up to 3jets: ud->HHud, uu->HHuu
- HHH+up to 2jets: ud->HHHud, uu->HHHuu

Ratio of Total Integration Time



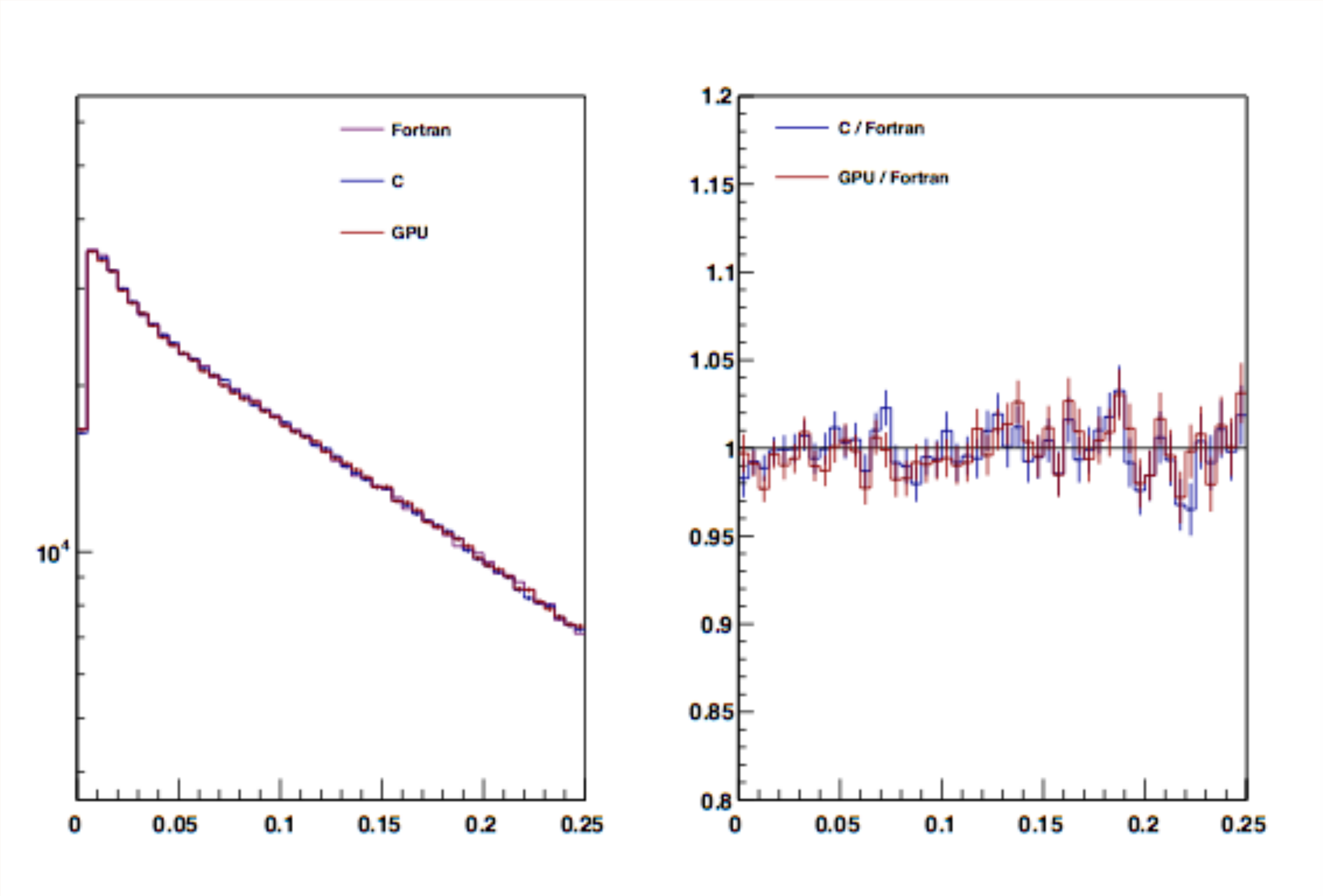
- Comparison of total execution time in double precision.

Event Generation by SPRING

- Generate unweighted events based on BASES results
- One thread generates one event in a certain hyper-cell of multi-dimension space (acceptance-rejection):
  -> the most inefficient hyper-cell determines the total process time
- Iterative reuse of threads:
  threads that have finished event generation can be assigned to inefficient hyper-cell at the next iteration
  -> improves total performance

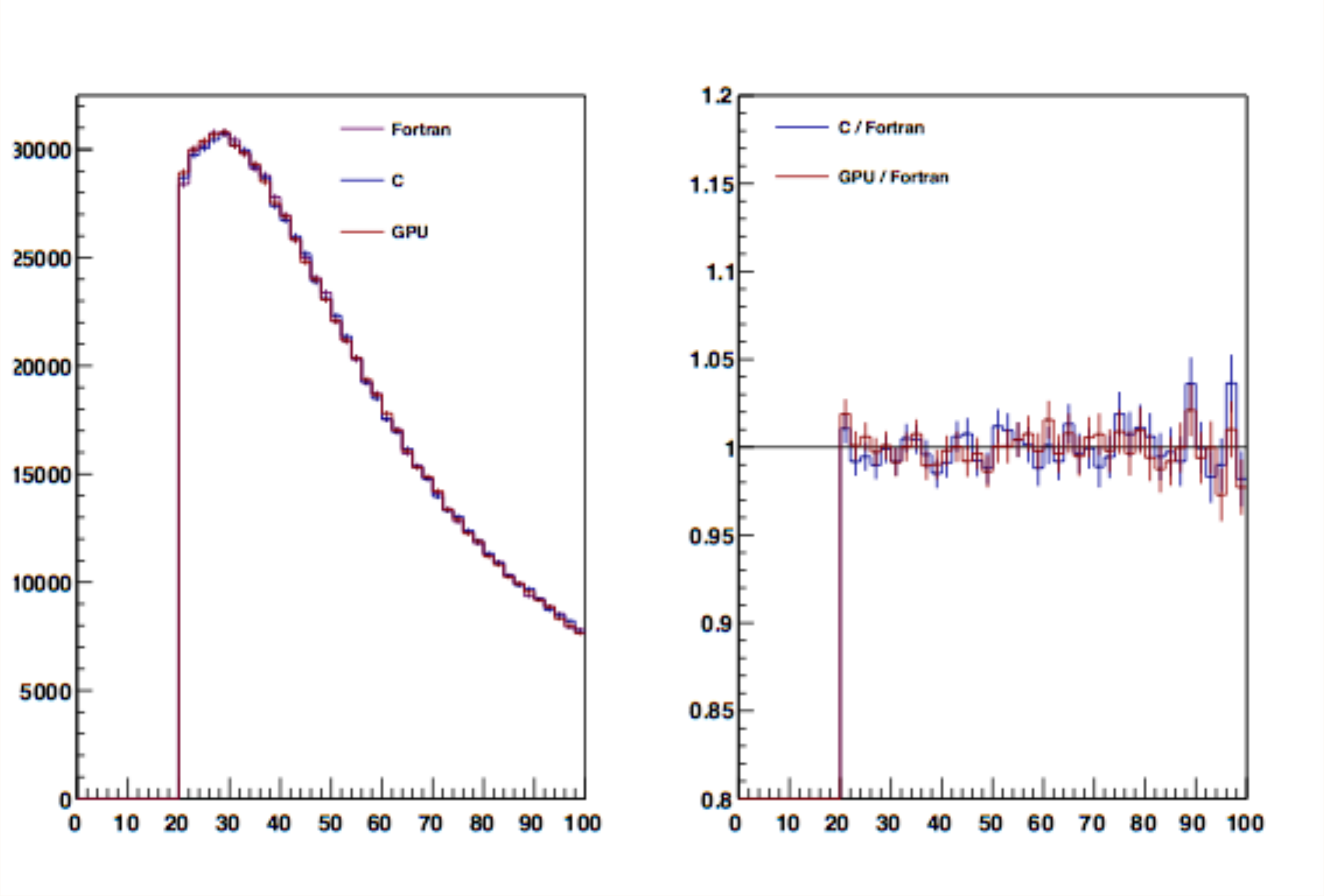
Generated distributions

- ud~ -> W^+ (->mu^+ vm) + 3-gluons (10^6 events).
- x1 (energy fraction of u):



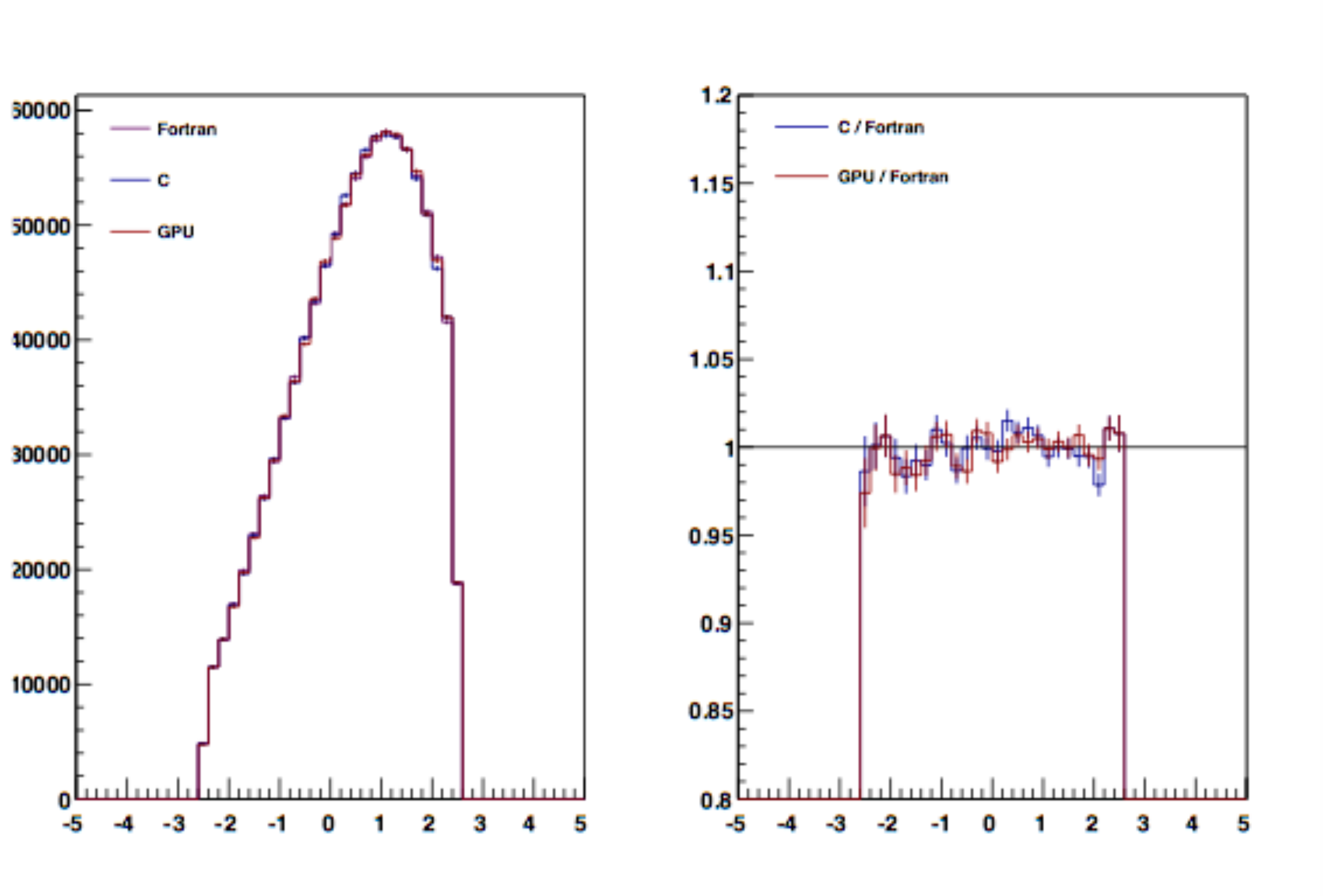
Generated distributions

- p_T (mu$^+$):



Generated distributions

- eta (mu^+):



SPRING performance

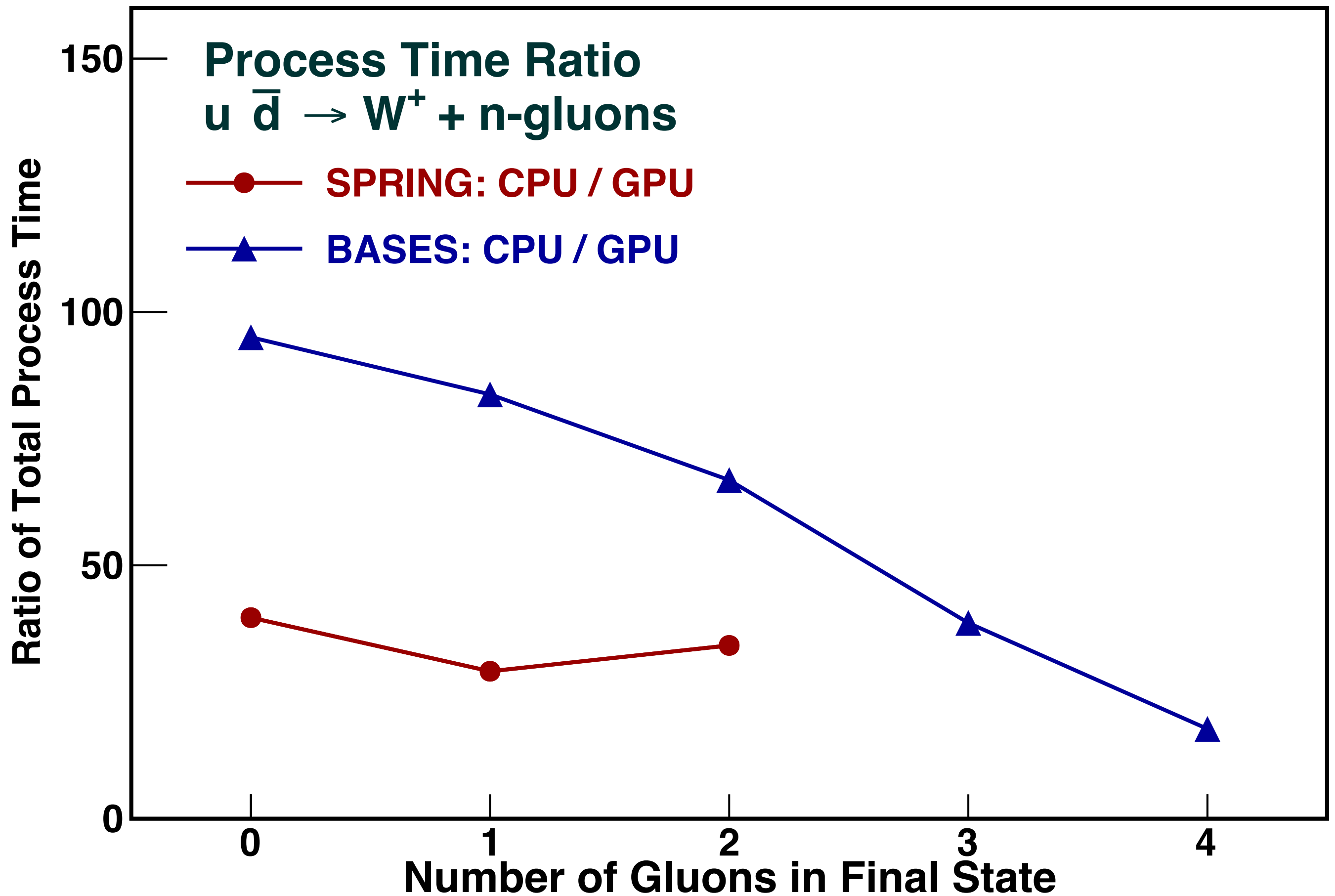
- Total execution time [sec]: generation of unweighted 10^6 events. Double precision results.

No. of gluons	FORTRAN	C2075	CPU/GPU
0	9.72	0.245	40
1	43.2	1.487	29
2	4224.8	123.6	34

More studies are necessary for processes with larger number of gluons.

Ratio of process time (C2075)

- Very preliminary results on C2075 in double precision.



Integration to MG5

HEGET for MG5

- Heget was developed on the bases of HELAS in FORTRAN.
- New “HEGET” package on GPU is based on C++ codes automatically generated by “ALOHA”.
  At present a few modifications are necessary.
  -> feedback for the GPU version of “ALOHA”
  -> Install to official “ALOHA” until the end of this year.
- Now preparing online interface for the generation of SM background processes.

MG5 on GPU

- Necessary parts for event generation on GPU are ready for use. But, still more works are necessary to provide an integrated system.
- Goal:
  - start from SM processes with arbitrary cuts on final states for background event generations.
  - provide interface like “Wjjjj”.
  - provide generation of downloadable code generation and hopefully online event generations.

Summary & Prospects

Summary & Prospect

- Program components of cross section computation and event generation based on MadGraph system can be executed on GPU with high performance:
  - GPU version of VEGAS and BASES/SPRING
- Improvement factor of performance can become between 10~100 for total execution time of BASES integration.
- Improvement of SPRING performance for muti-jets processes can be expected.

Summary & Prospect

- Integration to MG5 is now in progress.
  - will provide GPU codes for the generation of SM processes

- Hardware is improving and more applications of GPU to HEP software should be useful.
- Our products have been hosted with "madgraph.kek.jp". We lost the host due to hardware troubles, and now are trying to recover the site as soon as possible.